More on the Java Collections Framework

java.util.SortedSet (an interface)
- SortedSet extends Set
- For a SortedSet, the iterator() returns the elements in sorted order
- Methods (in addition to those inherited from Set):
  - public Object first();
  - Returns the first (lowest) object in this set
  - public Object last();
  - Returns the last (highest) object in this set
  - public Comparator comparator();
  - Returns the Comparator being used by this sorted set if there is one; returns null if the natural order is being used

java.lang.Comparable (an interface)
- public int compareTo(Object x);
- Returns a value (< 0), (= 0), or (> 0)
  - (< 0) implies this is before x
  - (= 0) implies this.equals(x) is true
  - (> 0) implies this is after x
- Many existing classes implement Comparable
  - String, Double, Integer, Char, java.util.Date…
  - If a class implements Comparable then that is considered to be the class’s natural ordering

java.util.Comparator (an interface)
- public int compare(Object x1, Object x2);
- Returns a value (< 0), (= 0), or (> 0)
  - (< 0) implies x1 is before x2
  - (= 0) implies x1.equals(x2) is true
  - (> 0) implies x1 is after x2
- Can often use a Comparator when a class’s natural order is not the one you want
  - String.CASE_INSENSITIVE_ORDER is a predefined Comparator
  - java.util.Collections.reverseOrder() returns a Comparator that reverses the natural order

SortedSet Implementations
- java.util.TreeSet
  - This is the only class that implements SortedSet
  - TreeSet’s constructors
    - public TreeSet();
    - public TreeSet(Collection c);
    - public TreeSet(Comparator comp);
    - public TreeSet(SortedSet set);
      (uses the same sorting order as that used by set)
- Write a method that prints out a SortedSet of words in order
- Write a method that prints out a Set of words in order

java.util.List (an interface)
- List extends Collection
- Items in a list can be accessed via their index (position in list)
- The add() method always puts an item at the end of the list
- The iterator() returns the elements in list-order
- Methods (in addition to those inherited from Collection)
  - public Object get(int index);
  - Returns the item at position index in the list
  - public Object set(int index, Object x);
  - Places x at position index, replacing previous item; returns the previous item
  - public void add(int index, Object x);
  - Places x at position index, shifting items to make room
  - public Object remove(int index);
  - Remove item at position index, shifting items to fill the space; returns the removed item
  - public int indexOf(Object x);
  - Return the index of the first item in the list that equals x (x.equals())
List Implementations

- java.util.ArrayList (an array; expands via array-doubling)
  - Constructors
    - public ArrayList();
    - public ArrayList(int initialCapacity);
    - public ArrayList(Collection c);
- java.util.LinkedList (a doubly-linked list)
  - Constructors
    - public LinkedList();
    - public LinkedList(Collection c);
  - Both include some additional useful methods specific to that class
  - Both are Cloneable

Efficiency Depends on Implementation

- Object x = list.get(k);
  - O(1) time for ArrayList
  - O(k) time for LinkedList
- list.remove(0);
  - O(n) time for ArrayList
  - O(1) time for LinkedList
- If (set.contains(x))…
  - O(1) expected time for HashSet
  - O(log n) for TreeSet

Summary

java.util.Map (an interface)

- Map does not extend Collection
- A Map contains key/value pairs instead of individual elements
- Methods
  - public Object put(Object key, Object value);
    - Associates value with key in the map; returns the old value associated with key or null if the key did not previously appear in the map
  - public Object get(Object key);
    - Returns the object to which this key is mapped or null if there is no such key
  - public boolean containsKey(Object key);
    - True iff Map contains a pair using the given key
  - public boolean containsValue(Object value);
    - True iff there is at least one pair with this value
  - public Object remove(Object key);
    - Removes any mapping for the key; returns old value associated with key if there was one (null otherwise)

More Map Methods

- Other methods
  - public int size();
    - Return the number of key/value pairs in the Map
  - public boolean isEmpty();
    - True iff Map holds no pairs
- Bulk methods
  - public void putAll(Map otherMap);
    - Puts all the mappings from otherMap into this map
  - public void clear();
    - Removes all mappings
- Sets/Collections derived from a Map
  - public Set keySet();
    - Returns a Set whose elements are the keys of this map
  - public Collection values();
    - Returns a Collection whose elements are all the values of this map
  - public Set entrySet();
    - Returns a Set of Map.Entry objects (can use getKey() and getValue())

java.util.SortedMap (an interface)

- Extends the Map contract: requires that keys are sorted
- The iterators for keySet(), values(), and entrySet() all return items in order of the keys
- Methods (in addition to those inherited from Map)
  - public Comparator comparator();
    - Returns the comparator used to compare keys for this map; null is returned if the natural order is being used
  - public Object firstKey();
    - Returns the first (lowest value) key in this map
  - public Object lastKey();
    - Returns the last (highest value) key in this map

...
Set and SortedSet Implementations

- **java.util.HashMap** (a class; implements Map)
  - Constructors
    - public HashMap ( );
    - public HashMap (Map map);
    - public HashMap (int initialCapacity);
    - public HashMap (int initialCapacity, float loadFactor);

- **java.util.TreeMap** (a class; implements SortedMap)
  - Constructors
    - public TreeMap ( );
    - public TreeMap (Map map);
    - public TreeMap (Comparator comp);
    - public TreeMap (SortedMap map);

Efficiency & Some Comments

- Both TreeMap and HashMap are meant to be accessed via keys
  - get, put, containsKey, remove are all fast
    - O(1) expected time for HashMap
    - O(log n) worst-case time for TreeMap
  - containsValue is slow
    - O(n) for both HashMap and TreeMap

- Both HashSet and TreeSet are actually implemented by building a HashMap and a TreeMap, respectively

The java.util.Arrays Utility Class

- Provides useful static methods for dealing with arrays
  - sort
    - Mostly uses QuickSort
    - Uses MergeSort for Object[] (it's stable)
  - binarySearch
  - equals
  - fill
- These methods are overloaded to work with
  - arrays of each primitive type
  - arrays of Objects

- Methods sort and binarySearch can use the natural order or there is a version of each that can use a Comparator
- There is also a method for viewing an array as a List: static List asList (Object[] a);
- Note that the resulting List is backed by the array (i.e., changes in the array are reflected in the List and vice versa)

The java.util.Collections Utilities

- public static Object min (Collection c);
- public static Object max (Collection c);
- public static Comparator reverseOrder ( ); // Reverse of natural order
- public static Comparator reverse (List list); // Reverse the list
- public static void shuffle (List list); // Randomly shuffle the list
- public static void fill (List list, Object x); // List is filled with x's
- public static void sort (List list); // Sort using natural order
- public static void sort (List list, Comparator comp);
- public static void binarySearch (List list, Object key);
- public static void binarySearch (List list, Object key, Comparator comp);

Unmodifiable Collections

- Dangerous version:
  - public final String suits[ ] = { "Clubs", "Diamonds", "Hearts", "Spades" };
- The final modifier means that suits always refers to the same array, but the array's elements can be changed
  - suits[0] = "Leisure";
- Safe version:
  - private final String theSuits[ ] = { "Clubs", "Diamonds", "Hearts", "Spades" };
  - public final List suits = Collections.unmodifiableList(Arrays.asList(theSuits));
- The Collections class provides unmodifiable wrappers; any methods that would modify the collection throw an UnsupportedOperationException
  - unmodifiableCollection, unmodifiableSet, unmodifiableSortedSet, unmodifiableList
  - unmodifiableMap, unmodifiableSortedMap

Summary